Tree Pruning With Subadditive Penalties

Clayton Scott, *Member, IEEE*

***Abstract*—In this paper we study the problem of pruning a binary tree by minimizing, over all pruned subtrees of the given tree, an objective function that combines an additive cost term with a penalty term that depends only on tree size. We present algorithms for general size-based penalties, although our focus is on subadditive penalties (roughly, penalties that grow more slowly than linear penalties with increasing tree size). Such penalties are motivated by recent results in statistical learning theory for decision trees, but may have wider application as well. We show that the family of pruned subtrees induced by a subadditive penalty is a subset of the family induced by an additive penalty. This implies (by known results about additive penalties) that the family induced by a subadditive penalty 1) is nested; 2) is unique; and 3) can be computed efficiently. It also implies that, when a single tree is to be selected by cross-validation from the family of prunings, subadditive penalties will never present a richer set of options than an additive penalty.**

***Index Terms*—Decision trees, nonadditive penalties, subadditive penalties, tree pruning.**

I. Introduction

TREE-BASED methods are one of the most widely applied techniques in all of applied mathematics and engineering, from nonparametric statistics and machine learning to source coding and multiscale signal and image processing [1]. In this paper, we focus on *pruning* trees via complexity regularization, a task that often occurs in the design of tree-based methods. Rather than focusing on a specific application or pruning problem, we present general results that apply in a number of different settings.

The formal statement of our problem is as follows. In graph theory a tree is a connected graph without cycles. We consider a specific kind of tree that we call a *rooted binary tree* which has the following properties:

1) there exists a unique node with degree 2;
2) all other nodes have degree 1 or 3.

The degree of a node is the number of edges linking that node to other nodes (called neighbors). The node with degree 2 is called the *root* node. Nodes with degree greater than one are called *internal* nodes and nodes with degree 1 are called *terminal* or *leaf* nodes. The *depth* of a node t is the number of edges traversed along the path between t and the root node. Every node t except the root has a *parent* which is the unique neighbor of t whose depth is one less than t's. Every internal node t has two *children* which are the two neighbors of t having depth one more than t's.

Manuscript received August 30, 2004. This work was supported by the National Science Foundation under Grants CCR-0310889 and CCR-0325571 and by the Office of Naval Research under Grant N00014-00-1-0966. The Associate Editor coordinating the review of this manuscript and approving it for publication was Dr. Kenneth E. Barner.

The author is with the Department of Statistics, Rice University, Houston, TX 77005 USA (e-mail: cscott@rice.edu).

Digital Object Identifier 10.1109/TSP.2005.859220

Rooted binary trees are readily envisioned by picturing the root node at the top of the graph and the remaining nodes "dangling down" in such a way that parents are above their children and one child branches left while the other branches right.

The set of leaf nodes of T is denoted $L(T)$. The *size* of a tree T is the number of leaf nodes and denoted $|T|$. A *subtree* of T is a subgraph $S \subseteq T$ that is a rooted binary tree in its own right. If S is a subtree that contains the root of T, we say S is a *pruned subtree* of T and write $S \preccurlyeq T$.

For the remainder of this paper, let T be a fixed rooted binary tree. Let ρ be a functional mapping subtrees of T to the positive reals. Let Φ be a mapping from the positive integers to the positive reals. We make the following assumptions on ρ and Φ.

1) ρ is monotonically nonincreasing, that is, $S_1 \preccurlyeq S_2 \Rightarrow \rho(S_1) \geq \rho(S_2)$.
2) ρ is *additive*, that is

$$\rho(S) = \sum_{t \in L(S)} \rho(t).$$

3) Φ is monotonically increasing, that is, $k_1 < k_2 \Rightarrow \Phi(k_1) < \Phi(k_2)$.

We are interested in algorithms computing and theorems describing two kinds of pruning problems. The first is

$$T^* = \arg\min_{S \preccurlyeq T} \rho(S) + \Phi(|S|). \tag{1}$$

If multiple trees achieve the minimum, choose T^* to be one with smallest size. Note that T^* is still not necessarily unique. The problem of solving (1) will be called *single pruning*, in contrast with *family pruning,* described below.

We refer to $\rho(S)$ and $\Phi(|S|)$ as the *cost* and *penalty* of S, respectively. Conceptually, every $S \preccurlyeq T$ is a model that explains some observed phenomenon. Typically $S = T$ is the most complicated model while the root node is the simplest. The idea behind pruning is to find a model that appropriately balances the complexity of S with the fidelity of S to an observed phenomenon.

One of the earliest and perhaps most widely known examples of this kind of pruning problem comes from the method of classification and regression trees (CART) of Breiman *et al.* [2]. In CART, a training dataset $(X_i, Y_i)_{i=1}^n$ is given, where X_i are feature vectors and the response variables Y_i satisfy $Y_i \in \{1, \ldots, M\}$ for classification and $Y_i \in \mathbb{R}$ for regression. The training data are used to construct an initial tree T that "overfits" the training data (for example, classifying every training sample correctly), and the purpose of pruning is to select a tree $S \preccurlyeq T$ that generalizes to accurately predict the correct Y for unlabeled X observed in the future.

For classification trees, each node $t \in T$ is assigned a class label y_t by majority vote over the training samples reaching t and $\rho(S)$ is taken to be the empirical error

$$\rho(S) = \frac{1}{n} \sum_{t \in L(S)} \sum_{i: X_i \in t} \mathbb{I}_{\{Y_i \neq y_t\}}.$$

Here $\mathbb{I}$ denotes the indicator function. For regression trees each $t \in T$ is assigned the empirical average (or perhaps some more general function of the samples in t)

$$y_t = \frac{1}{|\{i : X_i \in t\}|} \sum_{i: X_i \in t} Y_i$$

and $\rho(S)$ is the average empirical squared error

$$\rho(S) = \frac{1}{n} \sum_{t \in L(S)} \sum_{i: X_i \in t} (Y_i - y_t)^2.$$

For a penalty CART uses $\Phi(|S|) = \lambda |S|$ where $\lambda > 0$ is some constant. Additional examples of costs and penalties for tree structured source coding may be found in [3].

In many applications it is not known precisely how to calibrate ρ with respect to Φ so as to achieve an optimal pruning. In such cases it is customary to introduce a tuning parameter α, solve

$$T^*(\alpha) = \arg\min_{S \preccurlyeq T} \rho(S) + \alpha \Phi(|S|) \tag{2}$$

for several different values of α, and choose the best α by cross-validation. This is the second pruning problem we consider in this paper.

Since T is in general finite, it follows that there exist constants $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m = \infty$ and pruned subtrees $R_1, \ldots, R_m$ such that

$$\alpha \in [\alpha_{\ell-1}, \alpha_\ell) \Rightarrow T(\alpha) = R_\ell.$$

Because Φ is increasing, it follows that $|R_1| > \cdots > |R_m| = 1$, although these trees are not nested in general (see Section II for further discussion). We refer to $R_1, \ldots, R_m$ as the *family of prunings* of T with respect to ρ and Φ. The problem of computing these subtrees and thresholds is called *family pruning*. Note that in single pruning, no generality is gained by introducing a scalar multiplier α of Φ, for such a scalar may simply be absorbed into Φ.

A. Motivation

Single and family pruning have been studied extensively in the case where Φ is *additive,* by which we mean $\Phi(|S|) = \lambda |S|$ for some $\lambda > 0$ (for family pruning it suffices to take $\lambda = 1$). Additive penalties are by far the most popular choice for Φ, owing in large part to the existence of computationally efficient algorithms (which we review) for computing T^* and the family of prunings of T. Moreover, the family of prunings satisfies the desirable properties that the trees R_ℓ are unique and nested. In many cases, however, the choice of an additive penalty appears to have no other grounding besides computational convenience.

Several theoretical results, many of them recent, suggest that subadditive penalties may be more appropriate than additive penalties for certain applications. Roughly speaking, subadditive penalties are penalties that grow more slowly than additive penalties as a function of tree size (a precise definition is given in Section IV). For example, $\Phi(k) \propto k^\tau$ for $0 < \tau \leq 1$ defines a subadditive penalty. Barron [4] demonstrates risk bounds that, when applied to classification or regression trees, imply a penalty of $\Phi(k) \propto \sqrt{k}$. Mansour and McAllester [5], Nobel [6], and Scott and Nowak [7] also derive risk bounds for classification trees with $\Phi(k) \propto \sqrt{k}$. Mansour and McAllester [5] and Langford [8] derive penalties for classification trees that vary between $\Phi(k) \propto k$ and $\Phi(k) \propto \sqrt{k}$. Meanwhile, classification risk bounds for additive penalties are only known for the special "zero error" case (when the optimal classifier is correct with probability one) and under the more general but still quite restrictive "identifiability" assumption of Blanchard *et al.* [9]. In summary, subadditive penalties appear to have a much stronger theoretical foundation than additive ones in certain settings, especially classification. See [10] for further discussion.

Additive penalties naturally arise in estimation problems that employ a squared-error or L^2 loss function. For example, in regression, the squared bias grows linearly with degrees of freedom in a linear model. In classification, however, the probability of error can be expressed by an L^1 loss function. It thus seems reasonable to expect that nonadditive penalties would be more appropriate from a theoretical point of view for applications that use a non-L^2 loss function, such as source coding with an L^1 distortion.

B. Overview

The purpose of this paper is to present algorithms and relevant properties for single and family pruning with nonadditive, and in particular subadditive, penalties. One of our main results is that the family of prunings generated by a subadditive penalty is a subset of the family of prunings generated by the additive penalty. Positive implications of this fact are that subadditive families are nested and unique. It also leads to a simple algorithm for generating the family. A negative implication, however, is that when a tree is to be selected from the family of prunings by cross-validation (a very common strategy [2]), subadditive penalties never provide a richer class of options than the additive penalty.

This paper is organized as follows. In Section II, we study pruning with general size-based penalties. We give explicit algorithms for single pruning and family pruning and provide a geometric framework for interpreting the family of prunings. This section brings together several known results and perspectives, adds a few new insights, and sets the stage for our later discussion of subadditive penalties. In Section III, we review algorithms and properties related to pruning with additive penalties. In Section IV, we define dominating and subadditive penalties and prove a general theorem about nested families of prunings. We also explore in more detail the implications of this theorem as outlined above. Section V reports conclusions, including a discussion of possible extensions to other strategies for pruning trees.

II. General Sized-Based Penalties

We first present a general algorithm for single pruning when $\Phi(k)$ is arbitrary. This algorithm applies even if Φ is not necessarily increasing. The algorithm should not be considered novel;

its key components have appeared previously in other guises, as discussed below.

For each $k = 1, 2, \ldots, |T|$, define T^k to be a pruned subtree $S \preccurlyeq T$ (there may be more than one) minimizing $\rho(S)$ subject to $|S| = k$. These trees are referred to as *minimum cost trees*. Observe that

$$T^* = \arg\min\{\rho(T^k) + \Phi(k) : k = 1, 2, \ldots, |T|\} \quad (3)$$

is a solution to (1). In other words, it suffices to construct the sequence T^k and minimize the objective function over this collection. For the remainder of the paper, fix choices of T^k whenever T^k is not unique.

A. Computing Minimum Cost Trees

Since T is binary it has $|T|-1$ internal nodes. Let the nodes of T be indexed 1 through $2|T|-1$ in such a way that children have a larger index than their parents. (We refer to nodes and their indices interchangeably.) Let T_t denote the subtree rooted at node t and containing all of t's descendants in T (thus $T = T_1$). Let $l(t)$ and $r(t)$ denote the left and right children of node t, respectively. If U and V are pruned subtrees of $T_{l(t)}$ and $T_{r(t)}$, let $[[t, U, V]]$ denote the pruned subtree of T_t having U and V as its left and right subtrees, respectively. Finally, let T_t^k denote the pruned subtree of T_t having minimum cost among all pruned subtrees of T_t with k leaf nodes.

The algorithm for computing minimum cost trees is based on the following fact: if we know $T^i_{l(t)}$ and $T^j_{r(t)}$ for $i = 1, 2, \ldots, |T_{l(t)}|$ and $j = 1, 2, \ldots, |T_{r(t)}|$, it is a simple matter to find T_t^k, $k = 1, 2, \ldots, |T_t|$. For each $k = 1, 2, \ldots, |T_t|$, there exist i, j with $i + j = k$ such that $T_t^k = [[t, T^i_{l(t)}, T^j_{r(t)}]]$. This follows from additivity of ρ. Moreover, if $T_t^k = [[t, T^i_{l(t)}, T^j_{r(t)}]]$, then $\rho(T_t^k) = \rho(T^i_{l(t)}) + \rho(T^j_{r(t)})$. We may then set $T_t^k = [[t, T^{i^*}_{l(t)}, T^{j^*}_{r(t)}]]$, where i^*, j^* minimize $\rho(T^i_{l(t)}) + \rho(T^j_{r(t)})$ over all i, j such that $i + j = k$, $1 \le i \le |T_{l(t)}|$, and $1 \le j \le |T_{r(t)}|$. Note that i^*, j^* are determined by exhaustive search.

This step may be applied at each level of T, working from the bottom up, and leads to an algorithm for computing the minimum cost trees, and hence for determining T^*. The complete algorithm is presented in Fig. 1. The computational complexity of computing the minimum cost trees is $O(|T|^2)$. This was proved by Bohanec and Bratko [11]. The algorithm takes longer to run when T is more balanced. If T is maximally lopsided, e.g., all right children are terminal nodes, the algorithm computes the minimum cost trees in $O(|T|)$ operations.

The procedure described above for determining minimum cost trees is essentially the dual of the algorithm described in [11]. They considered the problem of finding the pruned subtree with smallest size among all pruned subtrees with empirical error below a certain threshold. This procedure was apparently known to the CART authors. As reported in [11], "Breiman ... did implement such an algorithm for optimal pruning; he was satisfied that it worked, but no further development was done, and the algorithm was not published." Subsequently, a somewhat more efficient implementation for the same problem was presented in [12]. As far as we know, this paper is the first to point out the use of minimum cost trees for single pruning with general size-based penalties.

Input:
  Initial tree T
Main Loop:
  For $t = 2|T| - 1$ downto 1
    Set $T_t^1 = \{t\}$;
    If t is not a terminal node, Then
      For $k = 2$ to $|T_t|$
        Set mincost $= \infty$;
        For $i = \max(1, k - |T_{r(t)}|)$ to $\min(|T_{l(t)}|, k - 1)$
          Set $j = k - i$;
          Set cost = $\rho(T^i_{l(t)}) + \rho(T^j_{r(t)})$;
          If cost $<$ mincost, Then
            Set mincost $=$ cost;
            Set $T_t^k = [\![t, T^i_{l(t)}, T^j_{r(t)}]\!]$;
          End If
        End For
      End For
    End If
  End For
Output:
  The minimum cost trees $T^k = T_1^k, k = 1, 2, \ldots, |T|$

Fig. 1. An algorithm for computing minimum cost trees. The limits for the innermost "For" loop ensure that i, j satisfy $i + j = k$, $1 \le i \le |T_{l(t)}|$, and $1 \le j \le |T_{r(t)}|$.

B. Geometric Aspects of Family Pruning

In this section, we introduce a geometric picture that leads to a general algorithm for family pruning with size-based penalties. To each $S \preccurlyeq T$, associate the function $f_S : [0, \infty) \to \mathbb{R}$ defined by $f_S(\alpha) = \rho(S) + \alpha\Phi(|S|)$. In this way each pruned subtree S maps to a line with y-intercept $\rho(S)$ and slope $\Phi(S)$, as shown in Fig. 2. Define

$$f^*(\alpha) = \min_{S \preccurlyeq T} f_S(\alpha).$$

Clearly f^* has the form

$$f^*(\alpha) = f_{R_\ell}(\alpha), \quad \alpha \in [\alpha_{\ell-1}, \alpha_\ell)$$

for some constants $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m = \infty$ and subtrees $R_\ell \preccurlyeq T$, $\ell = 1, 2, \ldots m$. Moreover, if Φ is monotonically increasing, $\Phi(|R_{\ell-1}|) > \Phi(|R_\ell|)$ implies $|R_{\ell-1}| > |R_\ell|$. These observations are summarized as follows.

Proposition 1: If $\Phi(k)$ is monotonically increasing in k, then there exist constants $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m = \infty$ and pruned subtrees $R_\ell \preccurlyeq T$, $\ell = 1, 2, \ldots m$, with $|R_1| > \cdots > |R_m| = 1$, such that $T(\alpha) = R_\ell$ whenever $\alpha \in [\alpha_{\ell-1}, \alpha_\ell)$.

This picture also provides us with an algorithm for determining the α_ℓ and R_ℓ. Observe that each R_ℓ must be a minimum cost tree T^k. Therefore

$$f^*(\alpha) = \min_k f_{T^k}(\alpha).$$

Clearly $R_1 = T^{k_1}$ where k_1 is the smallest k such that $\rho(T^k) = \rho(T)$. Now observe that, assuming $i < j$, f_{T^i} and f_{T^j} intersect at the point

$$\gamma_{i,j} = \frac{\rho(T^i) - \rho(T^j)}{\Phi(j) - \Phi(i)}.$$

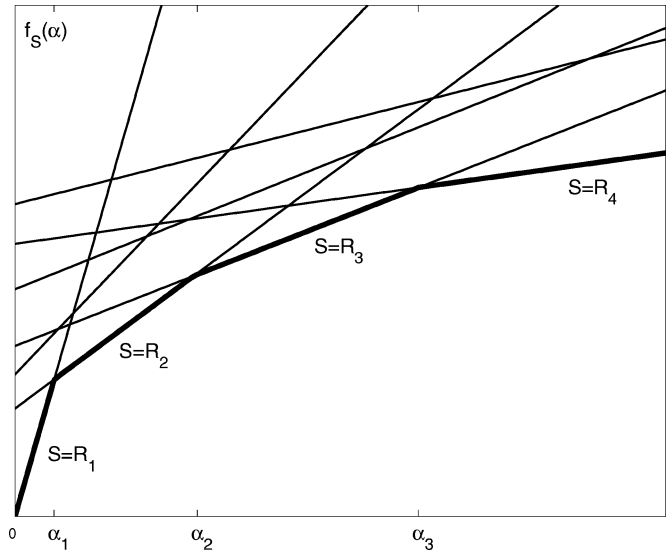


Fig. 2. Hypothetical plots of $f_S(\alpha) = \rho(S) + \alpha\Phi(|S|)$ as a function of α for all $S \preccurlyeq T$. Pruned subtrees coinciding with the minimum of these functions (shown in bold) over a range of α minimize the pruning criterion for those α.

Therefore, if $R_\ell = T^{k_\ell}$, then $R_{\ell+1} = T^{k_{\ell+1}}$, where

$$k_{\ell+1} = \arg\min_{k<k_\ell} \gamma_{k,k_\ell}.$$

If multiple k minimize the right-hand side, let $k_{\ell+1}$ be the smallest. Furthermore, we have

$$\alpha_\ell = \gamma_{k_{\ell+1},k_\ell}.$$

The algorithm is summarized in Fig. 3. This is the first algorithm of which we are aware for family pruning with a general size-based penalty.

We also highlight a property inherent in the definition of k_ℓ that will be of use later.

Lemma 1: If $k < k_\ell$, then $\gamma_{k_{\ell+1},k_\ell} \leq \gamma_{k,k_\ell}$. If $k < k_{\ell+1}$, then $\gamma_{k_{\ell+1},k_\ell} < \gamma_{k,k_\ell}$.

A second geometric picture due to Chou *et al.* [3], who consider only additive or affine penalties, offers essentially equivalent insights into the family of prunings of T. Consider the set of points $\mathcal{P} = \{p(S) = (\rho(S), \Phi(|S|)) | S \preccurlyeq T\} \subset \mathbb{R}^2$, as depicted in Fig. 4. The point corresponding to R_m (the root of T) is furthest down and to the right. The point corresponding to R_1 is furthest up and to the left (assuming $R_1 = T$). Moreover, the points corresponding to R_ℓ, $\ell = 1, \ldots, m$, are the vertices of the lower boundary of the convex hull of $\mathcal{P}$, listed counterclockwise. Thus, α_ℓ is the negative of the slope of the line segment connecting $p(R_\ell)$ to $p(R_{\ell+1})$. The algorithm described above for generating α_ℓ and R_ℓ can now be rederived in this setting by starting with R_1 and successively learning faces of the lower boundary of the convex hull of $\mathcal{P}$ in a counterclockwise fashion.

III. Additive Penalties

When $\Phi(|T|) = \lambda|T|$ for some $\lambda > 0$, there exist faster algorithms for single and family pruning than those described in the previous section. Moreover, the optimally pruned trees satisfy certain nice properties. The material in this section is taken from [2, ch. 10].

Input:
Minimum cost trees $T^k, k = 1, \ldots, |T|$
Initialization:
Set $k_1 = \arg\min\{k : \rho(T^k) = \rho(T)\}$;
Set $R_1 = T^{k_1}$;
Set $\ell = 1$;
Main Loop:
While $k_\ell > 1$
Set $\alpha_\ell = \infty$;
For $k = k_\ell - 1$ downto 1
Set $\gamma_{k,k_\ell} = (\rho(T^k) - \rho(T^{k_\ell}))/(\Phi(k_\ell) - \Phi(k))$;
If $\gamma_{k,k_\ell} \leq \alpha_\ell$
Set $\alpha_\ell = \gamma_{k,k_\ell}$;
Set $k_{\ell+1} = k$;
End If
End For
Set $\ell = \ell + 1$;
Set $R_\ell = T^{k_\ell}$;
End While
Output:
The family of prunings R_ℓ and thresholds α_ℓ.

Fig. 3. An algorithm generating the family of prunings and associated thresholds for an arbitrary increasing penalty.

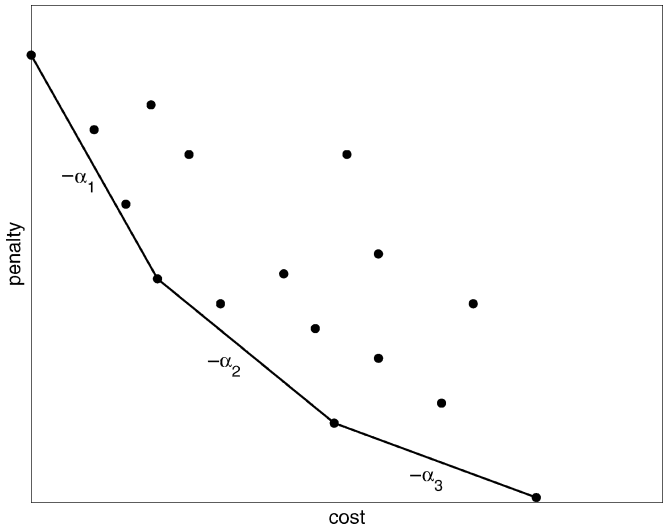


Fig. 4. Hypothetical plot of points $(\rho(S), \Phi(S))$ for all $S \preccurlyeq T$. The family of prunings corresponds to the vertices of the lower boundary of the convex hull, and the (negative) slopes between vertices correspond to the thresholds α_ℓ.

When λ is known, T^* may be computed by a simple bottom-up procedure. In particular, denoting

$$T_t^* = \arg\min_{S \preccurlyeq T_t} \rho(S) + \lambda|S|$$

we have $T_t^* = \{t\}$ for leaf nodes and for internal nodes

$$T_t^* = \arg\min\{\rho(S) + \lambda|S| : S = \{t\} \text{ or } S = [[t, T_{l(t)}^*, T_{r(t)}^*]]\}.$$

This last fact follows easily by additivity of ρ and Φ and by induction on t. This leads to an $O(|T|)$ algorithm for computing T^*, much faster than the more general $O(|T|^2)$ algorithm described previously. Moreover, T^* can be shown to be unique [2].

Breiman *et al.* [2] also prove the following theorem about the family of prunings generated by an additive penalty.

Theorem 1 (Breiman et al.): If $\Phi(k) = k$, then there exist weights $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m = \infty$ and subtrees $T \succcurlyeq$

$R_1 \succcurlyeq \cdots \succcurlyeq R_m = \{root\}$ such that $T(\alpha) = R_\ell$ whenever $\alpha \in [\alpha_{\ell-1}, \alpha_\ell)$.

In particular, the result is an improvement over Proposition 1 because the family of prunings is *nested*. We refer to the family $\{R_\ell\}$ in Theorem 1 as the CART trees.

The nesting property leads to an algorithm for finding these weights and subtrees [2], [3]. The algorithm has a worst case running time of $O(|T|^2)$, the worst case being when T is unbalanced. However, when T is balanced (i.e., when the depth of T is proportional to $\log|T|$, then the algorithm runs in time $O(|T|\log|T|)$. This yields an improvement (relative to the algorithm in Fig. 3) for many problems, such as those in signal and image processing, where the initial tree is often balanced.

The proof of Theorem 1 given by [2] is algebraic; an alternative geometric proof is given by [3]. These authors also extend the theorem to *affine* costs and penalties. A separate algebraic account may be found in [13].

IV. Sub-Additive Penalties

In this, the main section of this paper, we introduce subadditive penalties and show that for such penalties, the family of prunings is a subset of the CART trees. Thus, these trees are also unique, nested, and may be computed using the CART trees. To our knowledge, the results in his section are new, as is the following definition.

Definition 1: Let Φ^1 and Φ^2 be two increasing penalties. We say Φ^1 *dominates* Φ^2, denoted $\Phi^1 \gg \Phi^2$, if, for all positive integers $a > b > c$, we have

$$\frac{\Phi^2(a)-\Phi^2(b)}{\Phi^2(a)-\Phi^2(c)} \le \frac{\Phi^1(a)-\Phi^1(b)}{\Phi^1(a)-\Phi^1(c)}. \tag{4}$$

If $\Phi^1(k) = k$ and $\Phi^1 \gg \Phi^2$, we say Φ^2 is *subadditive*.

An important example of a subadditive penalty is the square root penalty $\Phi^2(k) = \sqrt{k}$. To see that this is indeed subadditive, observe that for $a > b > c$

$$\begin{aligned}\frac{\Phi^2(a)-\Phi^2(b)}{\Phi^2(a)-\Phi^2(c)} &= \frac{\sqrt{a}-\sqrt{b}}{\sqrt{a}-\sqrt{c}} \\ &< \frac{\sqrt{a}-\sqrt{b}}{\sqrt{a}-\sqrt{c}} \cdot \frac{\sqrt{a}+\sqrt{b}}{\sqrt{a}+\sqrt{c}} \\ &= \frac{a-b}{a-c} \\ &= \frac{\Phi^1(a)-\Phi^1(b)}{\Phi^1(a)-\Phi^1(c)}\end{aligned}$$

where $\Phi^1(k) = k$.

More generally, the following result characterizes a large class of penalties with $\Phi^1 \gg \Phi^2$.

Proposition 2: Let f, g be real valued, twice differentiable, increasing functions on (0, ∞), and for $k = 1, 2, \ldots$, set $\Phi^1(k) = f(k)$ and $\Phi^2(k) = g(k)$. Let x_0 be a positive real number. If $0 < g'(x) \le f'(x)$ and $g''(x) \le f''(x)[g'(x)/f'(x)]$ for all $x \ge x_0$, then (4) is satisfied for all real numbers $a > b > c \ge x_0$. Therefore, if $x_0 \le 1$, then $\Phi^1 \gg \Phi^2$.

Proof: The proof has two main steps. First, we prove the result for the special case $f(x) = x$; then we use the first step to establish the general case by reparameterizing $\mathbb{R}$.

Assume for now that $f(x) = x$. Then, by assumption, $g'(x) \le 1$ and $g''(x) \le 0$ for $x \ge x_0$. Let $a > b > c \ge x_0$. Note that (4) is equivalent to

$$\frac{\Phi^2(a)-\Phi^2(b)}{\Phi^2(b)-\Phi^2(c)} \le \frac{\Phi^1(a)-\Phi^1(b)}{\Phi^1(b)-\Phi^1(c)} \tag{5}$$

which can be seen by writing $\Phi^\kappa(a) - \Phi^\kappa(c) = (\Phi^\kappa(a) - \Phi^\kappa(b)) + (\Phi^\kappa(b) - \Phi^\kappa(c))$, for $\kappa = 1, 2$, and simplifying. Also note that Φ^1 and Φ^2 are monotonically increasing from the assumption on the first derivative. By the fundamental theorem of calculus, $\Phi^2(a) - \Phi^2(b) = \int_b^a g'(x)dx \le g'(b)(a-b)$, where we use the concavity of g in the last step. Similarly, $\Phi^2(b) - \Phi^2(c) = \int_c^b g'(x) \ge g'(b)(b-c)$. Summarizing, we have shown

$$\frac{\Phi^2(a)-\Phi^2(b)}{a-b} \le g'(b) \le \frac{\Phi^2(b)-\Phi^2(c)}{b-c}$$

which by (5) implies the theorem for this special case.

Now consider the general case. Define $\tilde{f}(x) = x$ and $\tilde{g}(x) = g(f^{-1}(x))$. Now $\tilde{f}'(x) = 1$, while $\tilde{g}'(x) = g'(f^{-1}(x))/f'(f^{-1}(x))$, which is ≤ 1 provided $x \ge \tilde{x}_0 := f(x_0)$. In addition, $\tilde{f}''(x) = 0$ and

$$\tilde{g}''(x) = \frac{g''(f^{-1}(x)) - f''(f^{-1}(x))\left[\frac{g'(f^{-1}(x))}{f'(f^{-1}(x))}\right]}{(f'(f^{-1}(x)))^2}$$

which is ≤ 0 if $x \ge \tilde{x}_0$. Thus, we may apply the previous case to $\tilde{f}$ and $\tilde{g}$. For all real numbers $x > y > z \ge f(x_0)$, we have

$$\frac{g(f^{-1}(x)) - g(f^{-1}(y))}{g(f^{-1}(y)) - g(f^{-1}(z))} \le \frac{x-y}{y-z}.$$

By taking $x = f(a)$, $y = f(b)$, and $z = f(c)$, and by monotonicity of f, we conclude

$$\frac{g(a)-g(b)}{g(b)-g(c)} \le \frac{f(a)-f(b)}{f(b)-f(c)}$$

which is what we wanted to show. ∎

The following corollary gives a concrete example of a family of penalties to which Proposition 2 applies.

Corollary 1: Let $\sigma, \tau > 0$, and set $\Phi^1(k) = k^\sigma$ and $\Phi^2(k) = k^\tau$. If $\sigma \ge \tau$, then $\Phi^1 \gg \Phi^2$. Hence, if $0 < \tau \le 1$, then Φ^2 is subadditive.

Proof: Define $f(x) = x^\sigma$ and $g(x) = x^\tau$. For $x \ge 1 = x_0$

$$g'(x) = \tau x^{\tau-1} \le \sigma x^{\sigma-1} = f'(x).$$

Furthermore, for $x \ge 1$

$$\begin{aligned}g''(x) &= \tau(\tau-1)x^{\tau-2} \\ &\le \tau(\sigma-1)x^{\tau-2} \\ &= \left(\sigma(\sigma-1)x^{\sigma-2}\right)\left(\frac{\tau x^{\tau-1}}{\sigma x^{\sigma-1}}\right) \\ &= f''(x)\left(\frac{g'(x)}{f'(x)}\right).\end{aligned}$$

Now apply Proposition 2. ∎

Further examples of dominating and subadditive penalties may be derived from the following result.

Proposition 3: If f, g, and x_0 satisfy the hypothesis of Proposition 2, then so do $\tilde{f}(x) = f(h(x))$, $\tilde{g}(x) = g(h(x))$, and $\tilde{x}_0 = h^{-1}(x_0)$ where $h : (0,\infty) \to (0,\infty)$ is any twice differentiable function such that $h'(x) > 0$ for all $x > 0$.

Proof: Observe that for any $x \geq \tilde{x}_0$

$$\begin{aligned}
0 < \tilde{g}'(x) &= g'(h(x)) \cdot h'(x) \leq f'(h(x)) \cdot h'(x) = \tilde{f}'(x) \\
\tilde{g}''(x) &= g''(h(x)) \cdot (h'(x))^2 + g'(h(x)) \cdot h''(x) \\
&\leq f''(h(x)) \left(\frac{g'(h(x))}{f'(h(x))} \right) \cdot (h'(x))^2 \\
&\quad + f'(h(x)) \left(\frac{g'(h(x))}{f'(h(x))} \right) \cdot h''(x) \\
&= \left(f''(h(x)) \cdot (h'(x))^2 + f'(h(x)) \cdot h''(x) \right) \\
&\quad \cdot \left(\frac{g'(h(x))}{f'(h(x))} \right) \\
&= \tilde{f}''(x) \left(\frac{\tilde{g}'(x)}{\tilde{f}'(x)} \right).
\end{aligned}$$

■

A. Main Result

For $\kappa = 1, 2$, and $\alpha \in \mathbb{R}$, define

$$T^{\kappa}(\alpha) = \arg\min_{S \preccurlyeq T} \rho(S) + \alpha \Phi^{\kappa}(|S|).$$

By Proposition 1, there exist scalars $\alpha_0 < \alpha_1 < \cdots < \alpha_m$ and $\beta_0 < \beta_1 < \cdots < \beta_n$, and subtrees $U_1, \ldots, U_m$ and $V_1, \ldots, V_n$, such that:

1) $\alpha \in [\alpha_{\ell-1}, \alpha_\ell) \Rightarrow T^1(\alpha) = U_\ell$;
2) $|U_1| > \cdots > |U_m| = 1$;
3) $\beta \in [\beta_{\ell-1}, \beta_\ell) \Rightarrow T^2(\beta) = V_\ell$;
4) $|V_1| > \cdots > |V_n| = 1$.

Theorem 2: With the notation defined above, if Φ^1 and Φ^2 are two increasing penalties such that $\Phi^1 \gg \Phi^2$, then $\{V_1, \ldots, V_n\} \subseteq \{U_1, \ldots, U_m\}$. In other words, for each β, there exists α such that $T^2(\beta) = T^1(\alpha)$.

An immediate application of the theorem is an alternate algorithm for pruning using a subadditive penalty. Let $\Phi^1(k) = k$ and let $R_1, \ldots, R_m$ denote the CART trees. These may be computed efficiently by the algorithm of [2] or [3]. By Theorem 2, if Φ^2 is subadditive, then $T^2(\beta)$ is one of these R_ℓ. Therefore

$$T^2(\beta) = \arg\min_{S \in \{R_1, \ldots, R_m\}} \rho(S) + \beta \Phi^2(|S|).$$

This last minimization may be solved by exhaustive search over the $m \leq |T|$ CART trees. We are unaware of a more direct way to do single pruning for subadditive penalties.

The theorem also implies a new algorithm for family pruning when Φ^2 is subadditive. The procedure is exactly like the one described in Fig. 3, except that one only needs to consider k (see line 3 of the main loop) such that

$$k \in \{i : i = |R_j| \text{ for some } j > \ell\}.$$

Thus it is not necessary to compute all minimum cost trees, only the CART trees, which can often be done more efficiently.

We have two distinct algorithms for computing the family of prunings induced by a subadditive penalty. Both algorithms have worst case running time $O(|T|^2)$. The first algorithm, discussed in Section II, is slower when T is more balanced, but prunes totally lopsided trees in $O(|T|)$ time. The second algorithm, just discussed, is slower when T is unbalanced, and runs in $O(|T| \log |T|)$ time when T is balanced. Conceivably, one could devise a test that determines how balanced a tree is in order to choose which of the two algorithms would be faster on a given tree.

Other properties for pruning with subadditive penalties follow from Theorem 2 and known results about the CART trees. For example, pruning with a subadditive penalty always produces unique pruned subtrees, and the family of pruned subtrees is nested.

Families of prunings are useful when the appropriate family member needs to be chosen by cross-validation. When this is the case, Theorem 2 implies that subadditive penalties will never provide a richer class of options than an additive penalty.

Finally, we remark that the proof of Theorem 2 only requires ρ to be nonincreasing, not necessarily additive. The theorem may also be of practical use in this more general setting.

B. Proof of Theorem 2

We require the following lemma. Recall that for $i < j$, we define

$$\gamma_{i,j} = \frac{\rho(T^i) - \rho(T^j)}{\Phi(j) - \Phi(i)}.$$

Lemma 2: Let a, b, c be positive integers with $a > b > c$. The following are equivalent:

1) $\gamma_{c,a} < \gamma_{c,b}$;
2) $\gamma_{b,a} < \gamma_{c,b}$;
3) $\gamma_{b,a} < \gamma_{c,a}$.

The three statements are also equivalent if we replace $<$ by $\leq$, $>$, $\geq$, or $=$.

Proof: A straightforward calculation establishes

$$\begin{aligned}
&(\Phi(a) - \Phi(c))(\Phi(b) - \Phi(c)) \left[\gamma_{c,a} - \gamma_{c,b}\right] \\
&\quad = (\Phi(a) - \Phi(b))(\Phi(b) - \Phi(c)) \left[\gamma_{b,a} - \gamma_{c,b}\right] \\
&\quad = (\Phi(a) - \Phi(b))(\Phi(a) - \Phi(c)) \left[\gamma_{b,a} - \gamma_{c,a}\right].
\end{aligned}$$

The lemma follows from these identities and the fact that Φ is increasing.

The lemma may also be established by geometric considerations. Consider the three points $p_a, p_b, p_c \in \mathcal{P}$ defined by T^a, T^b, T^c, respectively (see Section II-B). Note that p_a is above and to the left of p_b, which is above and left of p_c. Then $\gamma_{a,b}$ is the negative slope of the line segment connecting p_a and p_b, and similarly for the other two combinations of points. Then the statements in 1)–3) are all true if and only if p_b is strictly *above* the line connecting p_a and p_c. Similarly, all three statements hold with equality if and only if b lies *on* the line joining p_a and p_c, and so on. ■

To prove the theorem, first notice that $U_1 = V_1 =$ the smallest $S \preccurlyeq T$ such that $\rho(S) = \rho(T)$. The theorem follows by induction if we can show $V_j \in \{U_1, \ldots, U_m\} \Rightarrow V_{j+1} \in \{U_1, \ldots, U_m\}$. To show this, we suppose it is not true and arrive at a contradiction. Assume there exists j such that $V_j \in \{U_1, \ldots, U_m\}$ but $V_{j+1} \notin \{U_1, \ldots, U_m\}$. Then $V_j = U_i$ for some i. Moreover, there exists $k \geq 0$ such that $|U_{i+k}| > |V_{j+1}| > |U_{i+k+1}|$.

Introduce the notation

$$\gamma_{i,j}^{\kappa} = \frac{\rho(T^i) - \rho(T^j)}{\Phi^{\kappa}(j) - \Phi^{\kappa}(i)}$$

for $\kappa = 1, 2$, and $i < j$. Define $p = |U_i| = |V_j|$, $q = |U_{i+k}|$, $r = |V_{j+1}|$, and $s = |U_{i+k+1}|$. Then $p \geq q > r > s$. We will show

$$\gamma_{s,r}^2 \leq \gamma_{r,q}^2 \leq \gamma_{r,p}^2 < \gamma_{s,r}^2$$

thus arriving at our desired contradiction. Denote these inequalities by I1, I2, and I3, respectively.

To establish I1, observe

$$\begin{aligned}
\gamma_{s,q}^2 &= \frac{\rho(T^s) - \rho(T^q)}{\Phi^2(q) - \Phi^2(s)} \\
&= \frac{\rho(T^s) - \rho(T^q)}{\Phi^1(q) - \Phi^1(s)} \cdot \frac{\Phi^1(q) - \Phi^1(s)}{\Phi^2(q) - \Phi^2(s)} \\
&\leq \frac{\rho(T^r) - \rho(T^q)}{\Phi^1(q) - \Phi^1(r)} \cdot \frac{\Phi^1(q) - \Phi^1(s)}{\Phi^2(q) - \Phi^2(s)} \\
&= \frac{\rho(T^r) - \rho(T^q)}{\Phi^2(q) - \Phi^2(r)} \cdot \frac{\Phi^2(q) - \Phi^2(r)}{\Phi^1(q) - \Phi^1(r)} \cdot \frac{\Phi^1(q) - \Phi^1(s)}{\Phi^2(q) - \Phi^2(s)} \\
&\leq \frac{\rho(T^r) - \rho(T^q)}{\Phi^2(q) - \Phi^2(r)} \\
&= \gamma_{r,q}^2
\end{aligned}$$

where the first inequality follows from Lemma 1 and the second inequality comes from the definition of $\Phi^1 \gg \Phi^2$. Since $\gamma_{s,q}^2 \leq \gamma_{r,q}^2$, Lemma 2 (iii $\Rightarrow$ ii) implies $\gamma_{s,r}^2 \leq \gamma_{r,q}^2$, establishing I1.

To show I2, assume $p \neq q$ (otherwise the inequality is trivial). Note that Lemma 1 implies $\gamma_{r,p}^2 \leq \gamma_{q,p}^2$. Lemma 2 (iii $\Rightarrow$ i) then implies I2.

Finally, by Lemma 1, we have $\gamma_{r,p}^2 < \gamma_{s,p}^2$. I3 follows from Lemma 2 (iii $\Rightarrow$ ii). □

V. Conclusion

We have presented two polynomial time algorithms for pruning and generating families of prunings using nonadditive penalties. The first algorithm applies to arbitrary penalties, while the second algorithm applies to subadditive penalties. Both algorithms have a worst case run time of $O(|T|^2)$. The first algorithm achieves the worst case for balanced trees (i.e., when depth$(T) \propto \log |T|$) and only requires $O(|T|)$ operations for lopsided trees (e.g., when every left descendant is a leaf node). The second algorithm has its worst case when T is unbalanced, and runs in $O(|T| \log |T|)$ time for balanced trees.

The second algorithm is based on a general theorem that, as a special case, implies that the family of prunings induced by a subadditive penalty is a subset of the family induced by an additive penalty. This implies that subadditive families are unique and nested. It also implies a negative result: when cross-validation is to be used to select the best member from a family of prunings, subadditive penalties will never offer a richer set of options than an additive penalty. It does not imply, however, that subadditive penalties should never be used in conjunction with cross-validation. In principle it is possible that the extra trees gained by an additive penalty actually have poorer performance, in which case it is better to not even consider them as a possibility.

An immediate impact of this work is in the area of classification tree design. It has recently been shown that subadditive penalties are more appropriate than an additive penalty for pruning classification trees (as discussed in the introduction). The work presented here provides for efficient implementation of such strategies and characterizes the resulting pruned subtrees.

Future work may ask whether the results of this paper have analogues for other pruning strategies. Many alternatives to size-based pruning have been proposed (see [14]–[19] and references therein), and several of these methods contain tuning parameters analogous to the weight α, which in size-based pruning controls the tradeoff between cost and penalty. It would be interesting to know, for example: are there fast algorithms to compute the entire family of prunings as the tuning parameters vary, and does the family of pruned subtrees have any properties such as nestedness?

We briefly comment on two pruning strategies that may be of interest in this regard. Both are designed for the purpose of pruning classification trees. The first is called "pessimistic pruning" and refers to those algorithms that attempt to estimate the probability of error of a subtree based on its empirical error. The estimate is then used to decide whether to keep the subtree in a bottom-up pruning algorithm. Although pessimistic pruning has theoretical support [20], [21], the bounds used to motivate the selection criterion are often too loose in practice, and so the introduction of a tuning parameter (weight) becomes necessary. The second involves minimizing a sum of cost and penalty, but where the penalty is no longer a function simply of tree size. Recent developments in statistical learning theory suggest that penalties favoring *unbalanced* trees may lead to improved generalization error [5], [22], [19]. Again, however, these penalties are based on error bounds that are too loose in practice, and hence the introduction of a tuning parameter is again appropriate.

In conclusion, it is quite possible that other machine learning and signal processing tree-based methodologies employ an additive penalty simply for convenience, when perhaps a nonadditive penalty would be more appropriate. We hope this paper might lead to a reassessment of such problems.

Acknowledgment

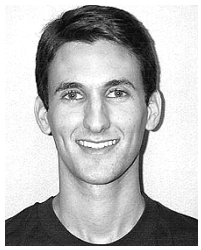
The author would like to thank R. Nowak and the anonymous reviewers for their feedback.

References

[1] S. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data Mining Knowledge Disc.*, vol. 2, no. 4, pp. 345–389, 1998.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

[3] P. Chou, T. Lookabaugh, and R. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Trans. Inform. Theory*, vol. 35, pp. 299–315, 1989.
[4] A. Barron, "Complexity regularization with application to artificial neural networks," in *Nonparametric Functional Estimation and Related Topics*, G. Roussas, Ed. Dordrecht, the Netherlands: Kluwer Academic, 1991, NATO ASI Series, pp. 561–576.
[5] Y. Mansour and D. McAllester, "Generalization bounds for decision trees," in *Proc. 13th Annu. Conf. Computational Learning Theory*, N. Cesa-Bianchi and S. Goldman, Eds., Palo Alto, CA, 2000, pp. 69–74.
[6] A. Nobel, "Analysis of a complexity based pruning scheme for classification trees," *IEEE Trans. Inform. Theory*, vol. 48, pp. 2362–2368, 2002.
[7] C. Scott and R. Nowak, "Dyadic classification trees via structural risk minimization," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, vol. 15.
[8] J. Langford, "Quantitatively tight sample complexity bounds," Ph.D. dissertation, Carnegie Mellon Univ., 2002.
[9] G. Blanchard, C. Schäfer, and Y. Rozenholc, "Oracle bounds and exact algorithm for dyadic classification trees," in *17th Annu. Conf. Learning Theory (COLT 2004)*, J. Shawe-Taylor and Y. Singer, Eds., Heidelberg, Jul. 1–4, 2004, pp. 378–392.
[10] C. Scott and R. Nowak, "Minimax optimal classification with dyadic decision trees," Rice Univ., Tech. Rep. TREE0403, 2004.
[11] M. Bohanec and I. Bratko, "Trading accuracy for simplicity in decision trees," *Machine Learn.*, vol. 15, pp. 223–250, 1994.
[12] H. Almuallim, "An efficient algorithm for optimal pruning of decision trees," *Artif. Intell.*, vol. 83, pp. 347–362, 1996.
[13] B. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[14] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[15] J. R. Quinlan and R. L. Rivest, "Inferring decision trees using the minimum description length principle," *Inform. Comput.*, vol. 80, no. 3, pp. 227–248, 1989.
[16] J. Mingers, "An empirical comparison of pruning methods for decision tree induction," *Machine Learn.*, vol. 4, pp. 227–243, 1989.
[17] F. Esposito, D. Malerba, and G. Semeraro, "A comparitive analysis of methods for pruning decision trees," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 5, pp. 476–491, 1997.
[18] E. Frank, "Pruning decision trees and lists," Ph.D. dissertation, Dept. of Computer Science, Univ. of Waikato, Hamilton, New Zealand, 2000.
[19] C. Scott and R. Nowak, "On the adaptive properties of decision trees," in *Advances in Neural Information Processing Systems*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, vol. 17.
[20] M. Kearns and Y. Mansour, "A fast, bottom-up decision tree pruning algorithm with near-optimal generalization," in *Proc. 15th Int. Conf. Machine Learning*, J. W. Shavlik, Ed., Madison, WI, 1998, pp. 269–277.
[21] Y. Mansour, "Pessimistic decision tree pruning based on tree size," in *Proc. 14th Int. Conf. Machine Learning*, D. H. Fisher, Ed., Nashville, TN, 1997, pp. 195–201.
[22] M. Golea, P. Bartlett, W. S. Lee, and L. Mason, "Generalization in decision trees and DNF: Does size matter?," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1998, vol. 10.



Clayton Scott (S'99–M'04) received the A.B. degree in mathematics from Harvard University, Cambridge, MA, in 1998 and the M.S. and Ph.D degree in electrical engineering from Rice University, Houston, TX, in 2000 and 2004.

He is currently a Postdoctoral Associate in Bioinformatics in the Department of Statistics, Rice University. His research interests include learning theory, bioinformatics, wavelets and multiscale analysis, and statistical signal and image processing.